

MODUL 4. Randomize, Defining Method, Collision Detection

Capaian Pembelajaran Praktikum:

- Membuat behavior random
- Mendefinisikan method baru
- Menerapkan collision detection di Greenfoot
- Membuat scenario interaktif dengan keyboard dan sound

Tools:

- Java Development Kit (JDK)
- Greenfoot IDE

Terminologi:

Isikan terminology yang sesuai untuk definisi dibawah ini:

[Defined Methods] A new method that a class didn't already possess; these methods are written in the class's source code below the act() method.

[Dot Notation] A technique that allows a class to use a method from another class or object. The dot between the class/object name and the method name indicates that the method comes from a different class or object.

[Constructor] A special kind of method that is automatically executed whenever a new instance of the class is created.

[new Keyword] A keyword that indicates that a new object is being created.

TRY IT / SOLVE IT:

1. Buat sebuah scenario baru. Buat subclass BeeWorld sebagai background dari scenario dengan ukuran lebar 700, tinggi 500 dan resolusi 1. (Gunakan image sand.jpg dari kategori backgrounds)

The screenshot shows a Java IDE window titled "BeeWorld - Main". The code editor displays the following Java code:

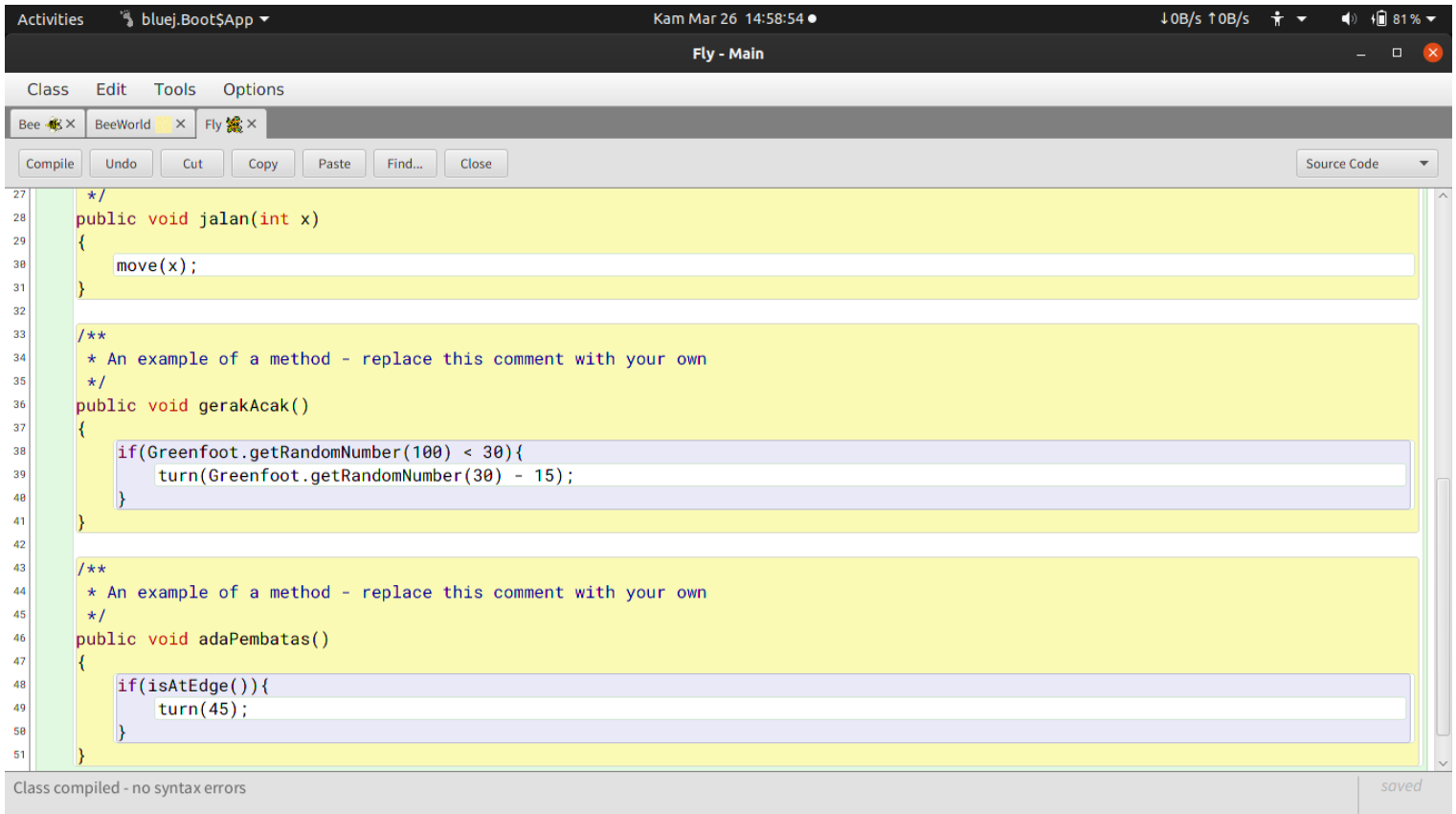
```

1 import greenfoot.*; // (World, Actor, GreenfootImage, Greenfoot and MouseInfo)
2
3 /**
4  * Write a description of class MyWorld here.
5  *
6  * @author (Adam Arthur Faizal)
7  * @version (26 Maret 2020)
8  */
9 public class BeeWorld extends World
10 {
11     /**
12      * Constructor for objects of class MyWorld.
13      *
14      */
15     public BeeWorld(){
16         // Create a new world with 700x500 cells with a cell size of 1x1 pixels.
17         super(700, 500, 1);
18         prepare();
19     }
20
21     /**
22      * Prepare the world for the start of the program.
23      * That is: create the initial objects and add them to the world.
24      */
25     private void prepare()

```

The IDE interface includes a menu bar (Class, Edit, Tools, Options), a toolbar (Compile, Undo, Cut, Copy, Paste, Find..., Close), and a status bar at the bottom indicating "Class compiled - no syntax errors" and "saved".

2. Tambahkan class Fly sebagai subclass Actor. Buatlah sebuah method *gerakAcak()* untuk Fly agar bergerak secara acak mengikuti kaidah sbb: Fly akan berbelok secara acak sampai dengan 15 derajat, dengan probabilitas 30%. Kemudian invoke atau call method *gerakAcak()* di method *act()*. (Gunakan image fly.png dari kategori animals)

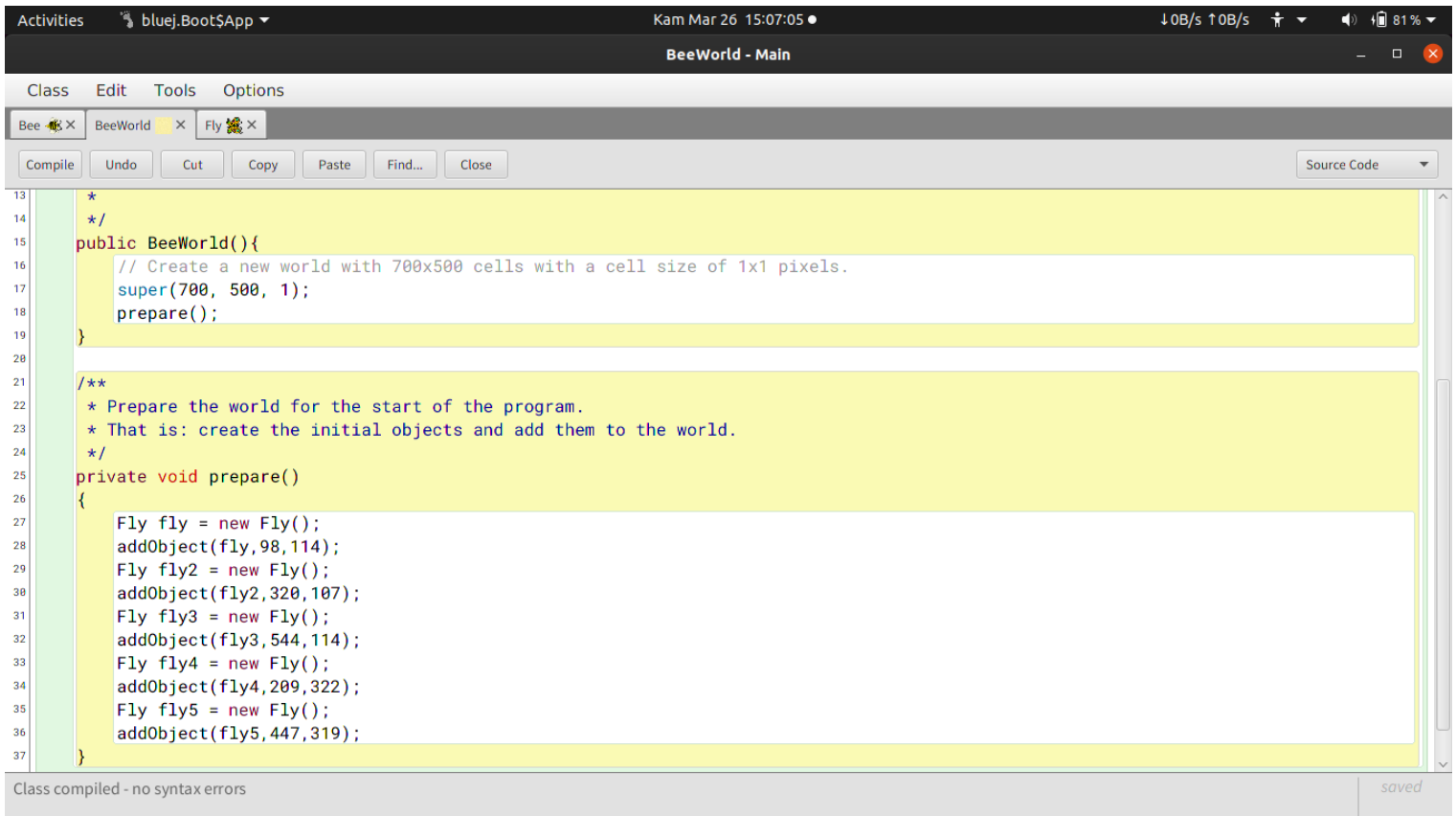


```

27  */
28  public void jalan(int x)
29  {
30      move(x);
31  }
32
33  /**
34   * An example of a method - replace this comment with your own
35   */
36  public void gerakAcak()
37  {
38      if(Greenfoot.getRandomNumber(100) < 30){
39          turn(Greenfoot.getRandomNumber(30) - 15);
40      }
41  }
42
43  /**
44   * An example of a method - replace this comment with your own
45   */
46  public void adaPembatas()
47  {
48      if(isAtEdge()){
49          turn(45);
50      }
51  }
    
```

Class compiled - no syntax errors

3. Dalam scenario yang anda buat, tambahkan 5 objek Fly kedalam BeeWorld secara otomatis dengan lokasi penempatan dibuat acak baik untuk koordinat X dan koordinat Y. (Gunakan method addObject untuk menambah objek & tambahkan kedalam konstruktor BeeWorld)

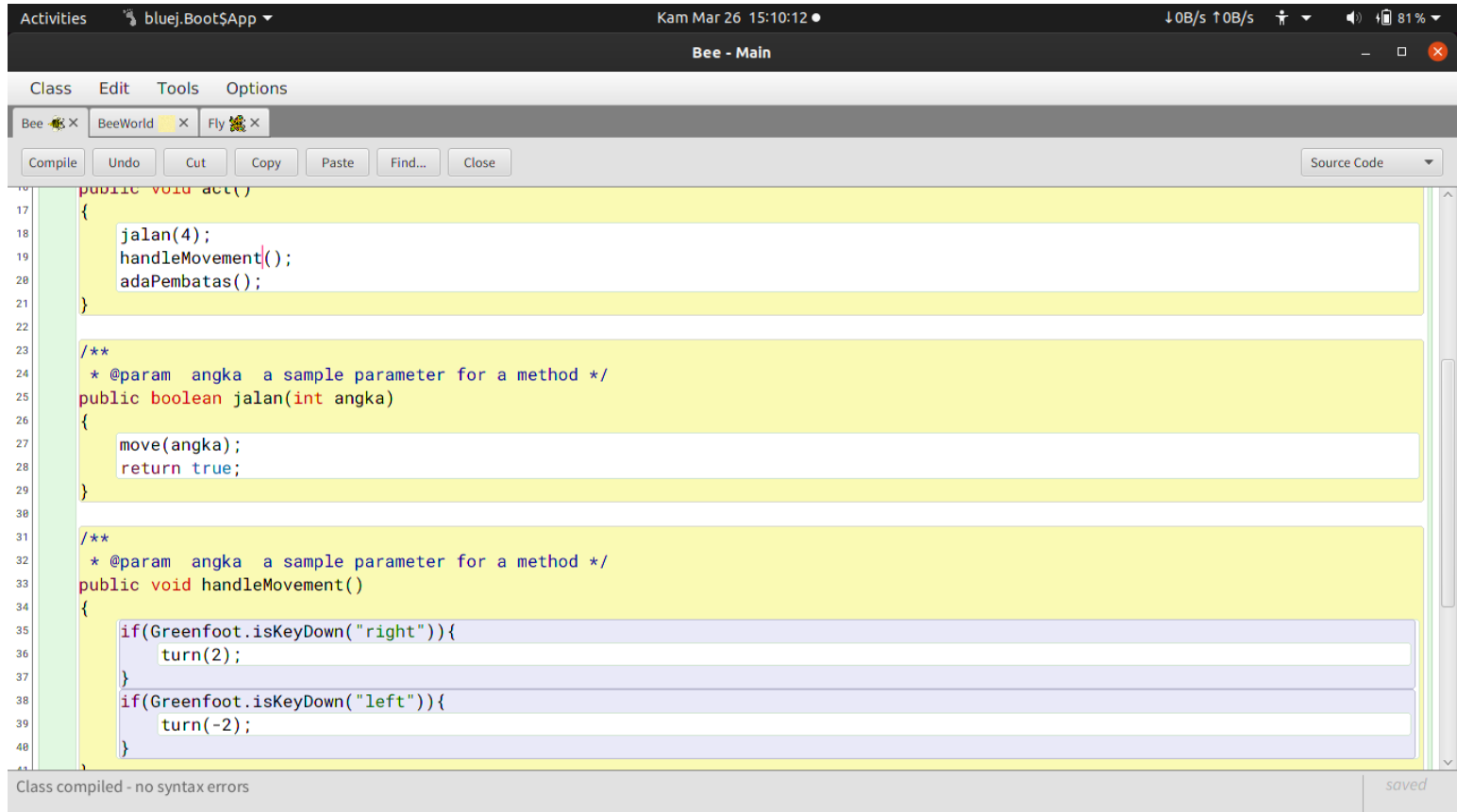


```

13  */
14  */
15  public BeeWorld(){
16      // Create a new world with 700x500 cells with a cell size of 1x1 pixels.
17      super(700, 500, 1);
18      prepare();
19  }
20
21  /**
22   * Prepare the world for the start of the program.
23   * That is: create the initial objects and add them to the world.
24   */
25  private void prepare()
26  {
27      Fly fly = new Fly();
28      addObject(fly, 98, 114);
29      Fly fly2 = new Fly();
30      addObject(fly2, 320, 107);
31      Fly fly3 = new Fly();
32      addObject(fly3, 544, 114);
33      Fly fly4 = new Fly();
34      addObject(fly4, 209, 322);
35      Fly fly5 = new Fly();
36      addObject(fly5, 447, 319);
37  }
    
```

Class compiled - no syntax errors saved

4. Buat class Bee dan buat method *handleMovement()* untuk menangani pergerakan Bee, yaitu Bee akan bergerak maju dengan kecepatan 2 dan akan berbelok kekiri sebesar 2 derajat ketika ada tombol left dari keyboard ditekan. Dan begitu pula sebaliknya, Bee akan berbelok ke kanan sebesar 2 derajat ketika tombol right dari keyboard ditekan. (Gunakan method *isKeyDown* dan statemen *if*)



```

16 public void act()
17 {
18     jalan(4);
19     handleMovement();
20     adaPembatas();
21 }
22
23 /**
24  * @param angka a sample parameter for a method */
25 public boolean jalan(int angka)
26 {
27     move(angka);
28     return true;
29 }
30
31 /**
32  * @param angka a sample parameter for a method */
33 public void handleMovement()
34 {
35     if(Greenfoot.isKeyDown("right")){
36         turn(2);
37     }
38     if(Greenfoot.isKeyDown("left")){
39         turn(-2);
40     }
41 }
    
```

Class compiled - no syntax errors

5. Buat method *eatFly()* di class Bee untuk menerapkan collision detection. Jika Bee berada dilokasi yang sama dengan objek Fly, maka objek Fly akan hilang / di remove dari BeeWorld. Kemudian objek Fly baru akan ditambahkan kedalam BeeWorld secara acak dengan jarak minimal 10 dari tepi. Panggil method yang anda buat di method *act()*.

The screenshot shows a Java IDE window titled "Bee - Main". The code editor displays the following Java code:

```

36     turn(5);
37 }
38 if(Greenfoot.isKeyDown("left")){
39     turn(-5);
40 }
41 }
42
43 /**
44  * An example of a method - replace this comment with your own
45  *
46  */
47 public void eatFly()
48 {
49     Fly fly = (Fly) getOneIntersectingObject(Fly.class);
50     if(fly != null){
51         getWorld().removeObject(fly);
52         getWorld().addObject(fly, Greenfoot.getRandomNumber(700), Greenfoot.getRandomNumber(500));
53         fly.setRotation(360);
54     }
55 }
56
57 public void atTepi()
58 {
59     if(getX() >= getWorld().getWidth()){
60         setLocation(0, getY());

```

The IDE interface includes a menu bar (Class, Edit, Tools, Options), a toolbar (Compile, Undo, Cut, Copy, Paste, Find..., Close), and a status bar at the bottom indicating "Class compiled - no syntax errors" and "saved".

6. Tambahkan method *atTepi()* di class Bee, yaitu ketika objek Bee berada di tepi kanan maka Bee akan berpindah ke tepi sebelah kiri. Jika Objek Bee berada di tepi atas maka objek Bee akan berpindah ke tepi bawah. Begitu pula sebaliknya.

The screenshot shows a Java IDE window titled "Bee - Main". The code editor displays the following Java code:

```

49 Fly fly = (Fly) getOneIntersectingObject(Fly.class);
50 if(fly != null){
51     getWorld().removeObject(fly);
52     getWorld().addObject(fly, Greenfoot.getRandomNumber(700), Greenfoot.getRandomNumber(500));
53     fly.setRotation(360);
54 }
55
56
57 public void atTepi()
58 {
59     if(getX() >= getWorld().getWidth()){
60         setLocation(0, getY());
61     }
62     if(getY() >= getWorld().getHeight()){
63         setLocation(getX(), 0);
64     }
65     if(getX() == 0){
66         setLocation(getWorld().getWidth(), getY());
67     }
68     if(getY() == 0){
69         setLocation(getX(), getWorld().getHeight());
70     }
71 }
72
73

```

The code implements the *atTepi()* method for the Bee class. It checks the current position of the bee against the world's width and height. If the bee is at the right edge, it moves to the left edge. If it's at the bottom edge, it moves to the top edge. If it's at the left edge, it moves to the right edge. If it's at the top edge, it moves to the bottom edge. The code uses *getWorld().getWidth()*, *getWorld().getHeight()*, *getX()*, *getY()*, and *setLocation()* methods.

At the bottom of the IDE, a status bar indicates "Class compiled - no syntax errors" and a "saved" button is visible.

Setelah sesi praktikum SELESAI, laporan praktikum dan source code (zip) harus dikirim/diupload ke google classroom pada hari yang sama.